ChatGPT Internals for coders

Jakub Švehla

Goals

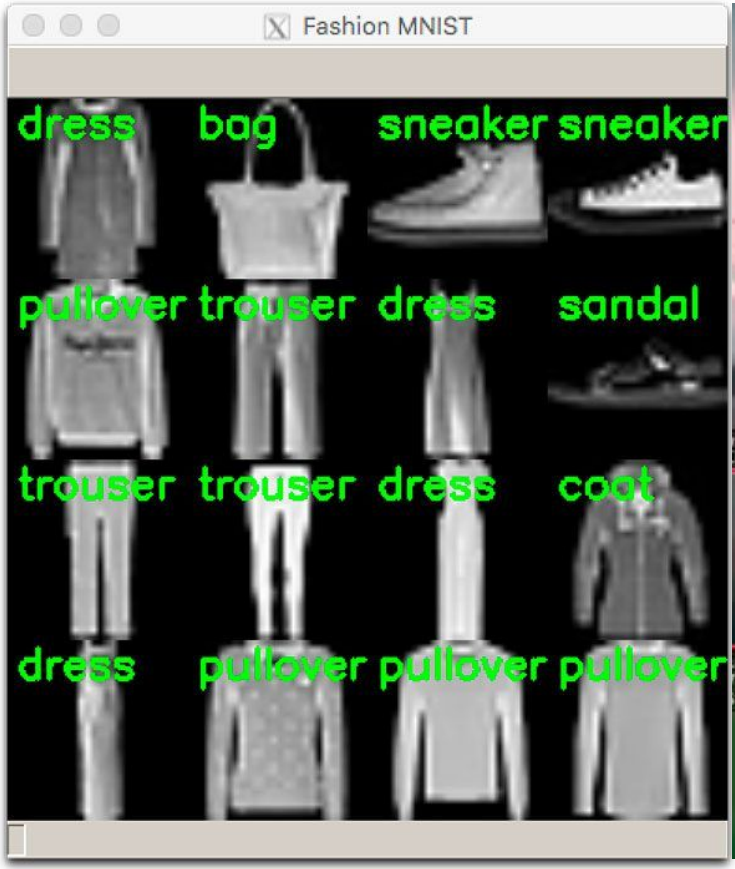
1. What is generative AI?
2. Understand how chatGPT is implemented
3. Current challenges in the world of Natural Language Processing
4. Ezoteric future chatGPT predictions

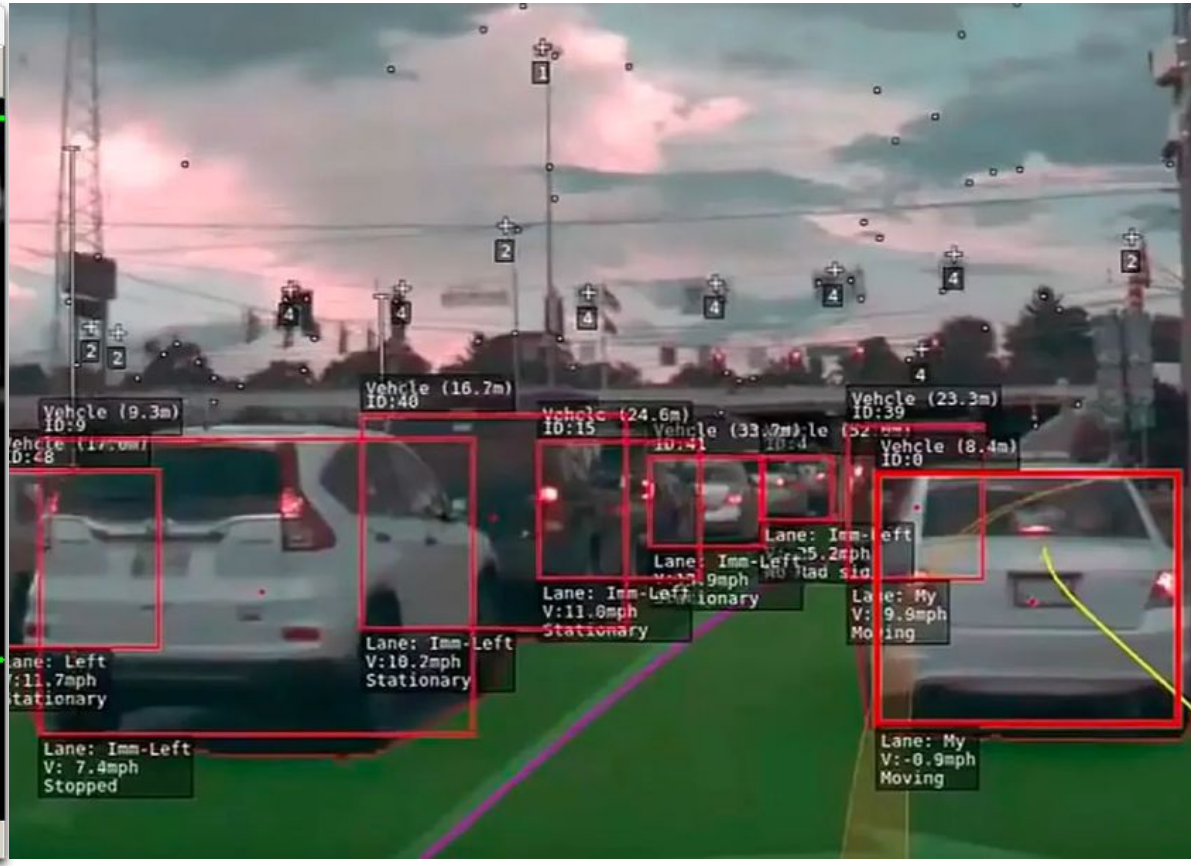
What is generative AI?

AI

- Image Classifiers
- User Clustering & auto segmentation in Online Advertising
- Self-organisation & self-assemblings
- Generative models
  - Images
  - 3D models
  - Texts (NLP - natural language processing)
- …

Classifiers





Self assembling AI

https://sebastianrisi.com/self_assembling_ai/

Generative AI - Images

Generative AI - 3D objects

Generative AI - Natural language processing

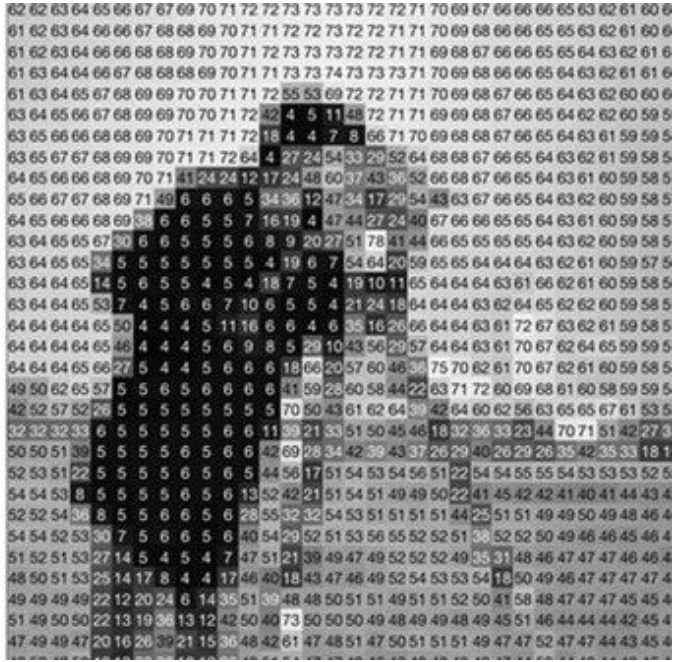
```
// write a binary search algorithm
const binarySearch = (arr, target) => {
    let left = 0;
    let right = arr.length - 1;
    let middle = Math.floor((left + right) / 2);
    while (arr[middle] !== target && left <= right) {
        if (target < arr[middle]) {
        right = middle - 1;
        } else {
        left = middle + 1;
        }
        middle = Math.floor((left + right) / 2);
    }
    return arr[middle] === target ? middle : -1;
    }
```

write a story about myself working in Bloomreach

Once upon a time, there was a talented young professional named Alex who dreamed of working for a leading tech company that would challenge and inspire him every day. After applying to numerous positions and undergoing a rigorous interview process, Alex was thrilled to receive an offer from Bloomreach, one of the most innovative and dynamic companies in the tech industry.

From day one, Alex was impressed by the culture of Bloomreach, which fostered collaboration, creativity, and continuous learning. As a member of the product development team, Alex was tasked with designing and implementing new features for Bloomreach's groundbreaking search and merchandising platform. Although the work was challenging,

How computers see images



How computers see text

Computers are able to `/` `*` `+` `-`

Can't do math between those number

Hello Bloomreach!	**72 101 108 108 111 32 66 108 111 111 109 114 101 97 99 104 33 10**
where is the closest pizza place	**119 104 101 114 101 32 105 115 32 116 104 101 32 99 108 111 115 101 115 116 32 112 105 122 122 97 32 112 108 97 99 101**

Reimplement ChatGPT from scratch

LLM (little language model)

(Chat Little Generative pretrained transformerish)

1. Download Corpus
2. Tokenizer
3. One hot encoding
4. Word Embeddings
5. Contextual Word Embeddings
6. Next word prediction
7. Fine tuning for Instruct GPT & ChatGPT & AI safeness
8. Scaling hardware

LLM Corpus

```
Bloomreach Loves Bloomreach

Loves!!!!

Bloomreach!!!

Loves Bloomreach

Bloomreach Loves Bloomreach !!!

! Bloomreach Loves Bloomreach !
```

Tokenizer

Tokenizer

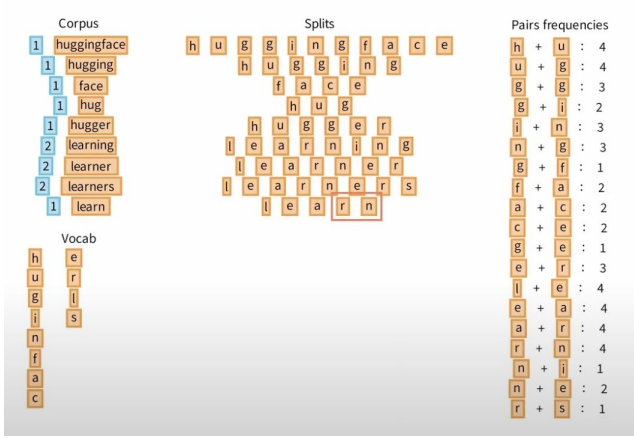
- Split corpus into smaller tokens (f.e: words)
- How many possible tokens should we support?
- Compute tokens from our text corpus

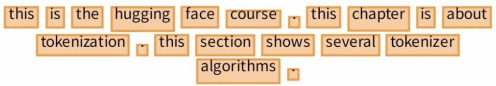
Tokenizer problematic exceptions

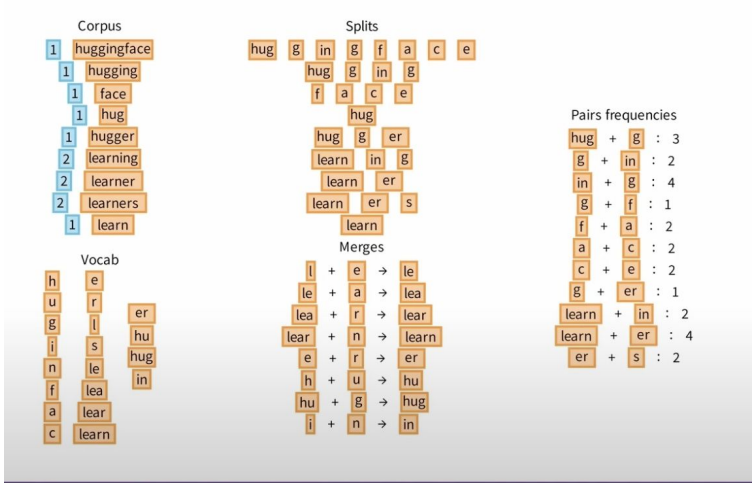
Typos	bloormeach
Case sensitive	BLOOMREACH → [BLOOMREACH] | [bloomreach]
Numbers	1234 → [1 2 3 4] | [1234]
Abbreviation	U.S.A. → [U . S . A .] | [U.S.A.]
idiomatic expressions	PCI Express → [“PCI Express”] | [“PCI” “Express”]
New Words	GPT → [GPT] | [G P T]
说/**說**	说/**說** → [说 / **說**]

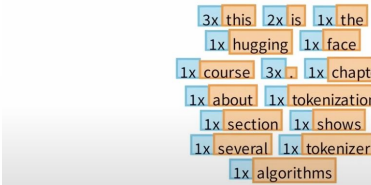
Tokenizer - BPE

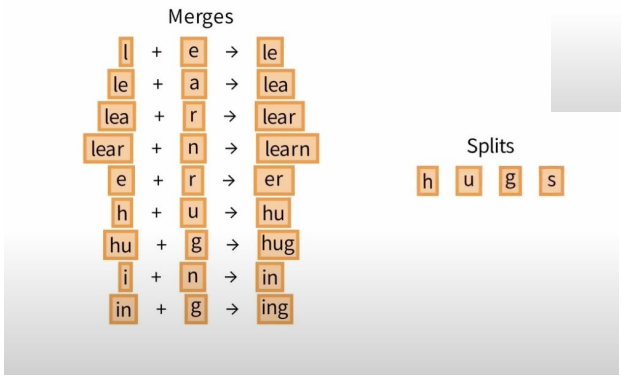
- Compute probabilities (relevancy) of sequences
- https://platform.openai.com/tokenizer (tik-token)











Bloomreach	Loves	Bloomreach			
Loves	!	!	!	!	
Bloomreach	!	!	!		
Loves	Bloomreach				
Bloomreach	Loves	Bloomreach	!	!	!
!	Bloomreach	Loves	Bloomreach	!	

One Hot Encoding

One Hot Encoding

- Transform a set of “non continuous” data into 2D points
- Create a new dimension for each token
- Easy for human readable and setting the data

(Bloomreach, Loves, !)						
[2, 1, 0]	Bloomreach	Loves	Bloomreach			
[1, 1, 4]	Loves	!	!	!	!	
[1, 0, 3]	Bloomreach	!	!	!		
[1, 1]	Loves	Bloomreach				
[2, 1, 3]	Bloomreach	Loves	Bloomreach	!	!	!
[2, 1, 2]	!	Bloomreach	Loves	Bloomreach	!	

```
lovely(sentence1) > lovely(sentence3)
lovely([2, 1, 0]) > lovely([1, 0, 3])
                1 > 0 = true
```

One Hot Encoding

Props

- Simple to create
- Possible to do arithmetics
- Simple to understand
- Used by many algorithms like
  - BM25 / TF-IDF (term frequency–inverse document frequency)

Cons

- Can’t recognize synonyms opposite words
- In the vectors we do not have meaning
- Gigantic count of dimensions (tens of thousands)
- The more tokens we have the more dimensions we need to represent basic words.

INTERRUPTION!!!
Math basics

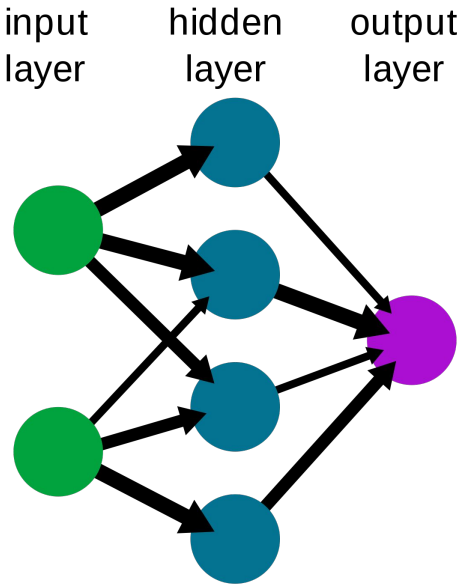
Math int - Math operations

1. Vector
2. Vector space
3. Linear transformation
4. Non-linear transformation
5. Neural networks

Math int - Neural networks

A program that helps to find the best matrices that we will apply to our data to approximate general-purpose behavior.

A simple neural network



Word embeddings

Word Embeddings

- Dimension-reduced One hot encoded vector space (latent space)
- Vectorized tokens with meanings in the latent space
- The latent space may be multilingual
- Each dimension describes some feature of the data

Word Embeddings - training

- Word2Vec + GloVE (most famous algorithms)
- Unsupervised learning on the giants corpuses
- What does word mean
- Looking for the most universal linear transformation
  - lower embedding dimensions
  - Middle word masking

Word Embeddings

Props

- Lookup table O(1) runtime complexity
- Can do arithmetics
- Measure similarity
  - `dot product`
  - `cosine similarity`
  - `euclidean distance`)
- MLM (multi languages models) Transfer learning
  - Abstraction over natural languages (`vec(car) ~ vec(auto)`)
  - F.e.: Need training data in english to learn some stuffs in czech

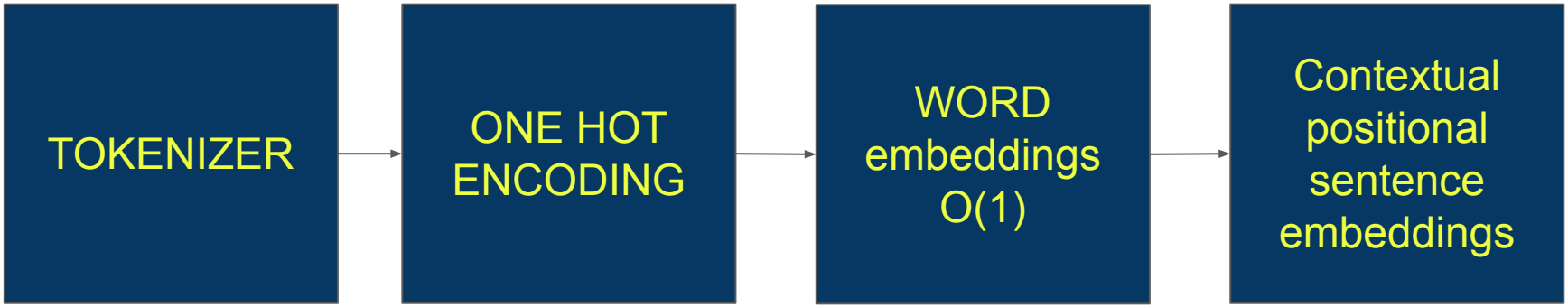
Cons

- No context word by meaning
- No sentence embeddings
- No positional encoding


TOKENIZER
ONE HOT ENCODING
WORD embeddings O(1)

Contextual Word/Sentence embeddings

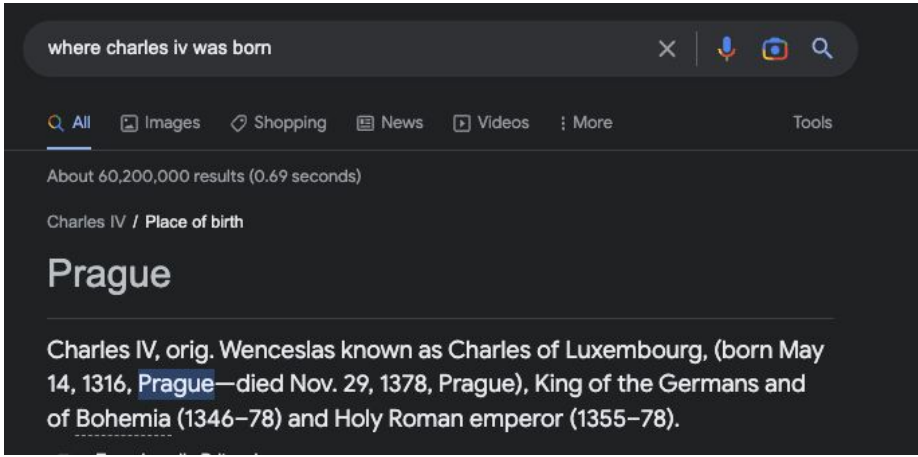
- Create a new linear transformation to transform word embeddings to produce
  - Contextual embeddings
  - Positional embeddings
  - Sentence embeddings

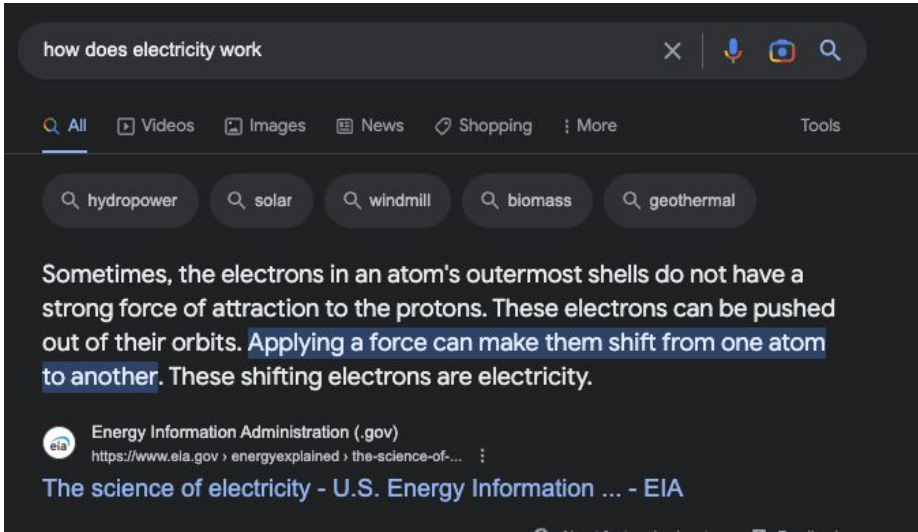

TOKENIZER
ONE HOT ENCODING
WORD embeddings O(1)
Contextual positional sentence embeddings

Fine-tune Embeddings

Fine tuning

- Where / how do we use embeddings?
- Fine tuning M input => 1 output
  - Google search
  - Question answering
  - Sentiment analysis
  - Toxicity classification
  - Automatic data annotation
- Compare similarity
  - Using vector similarity
  - “I like movies” ~ “cinemas are awesome”





Generative models

—— with naive bayes ——

M words on input → N words on output

- Translators
- Summary text
- ChatGPT

GPT LLM (generative pretrained transformer)

Predict next word

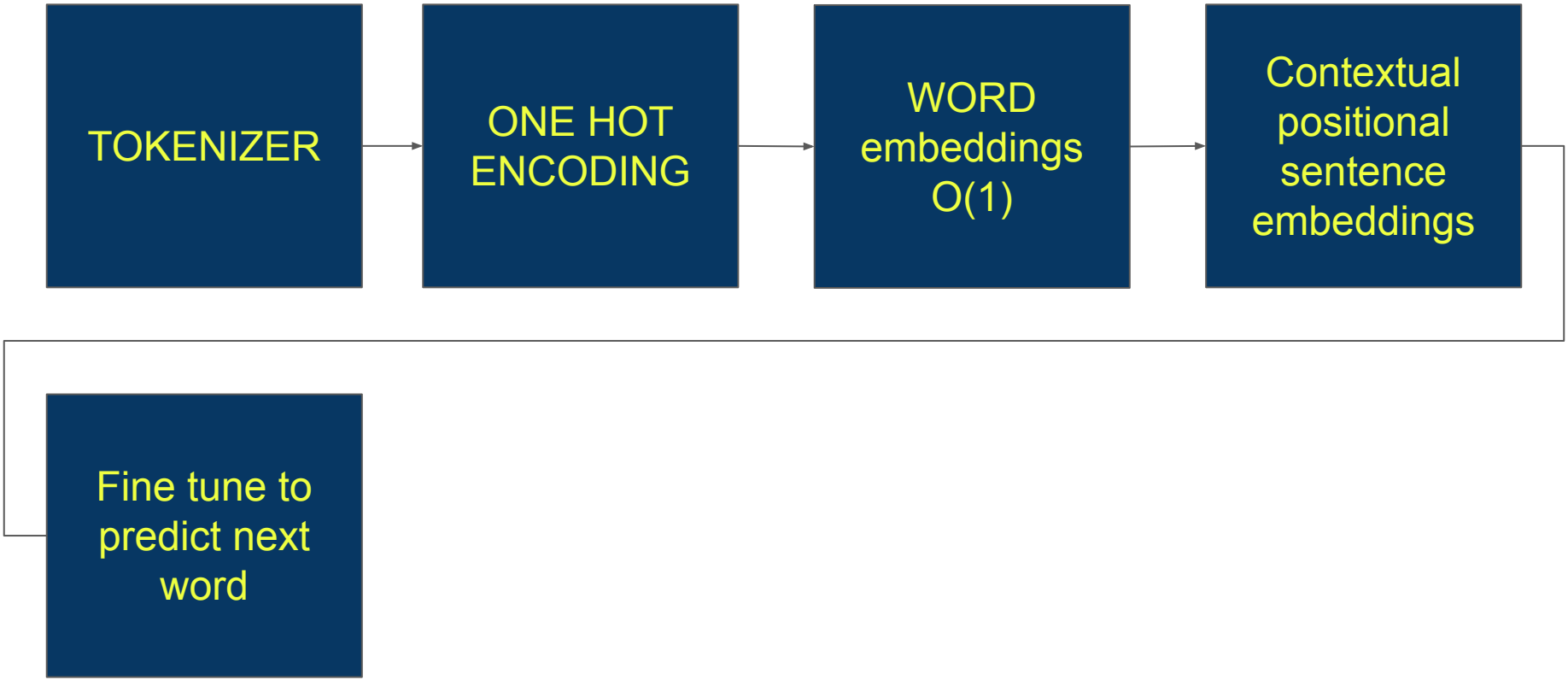
Apply auto-regressive loop

Based on probabilities, space transformations may generate something new

Setup Temperature with naive bayes

Hello, how are you	→	my
Hello, how are you my	→	friend
Hello, how are you my friend	→	?

Bloomreach is my	→	favorite
Bloomreach is my favorite	→	company
Bloomreach is my favorite company	→	🤗

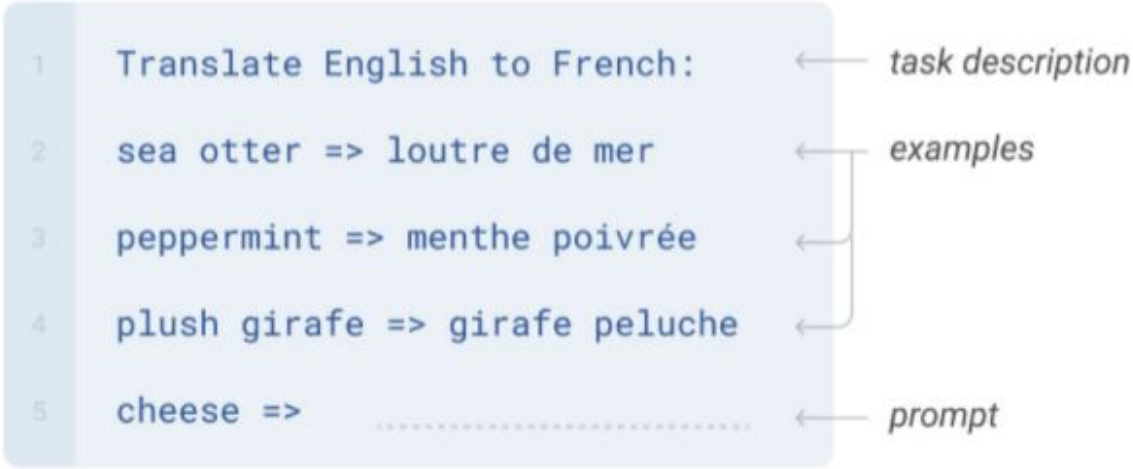

TOKENIZER
ONE HOT ENCODING
WORD embeddings O(1)
Contextual positional sentence embeddings
Fine tune to predict next word

Fine-tune LLM in chatLGPT

chatGPT *fine tuning*

Zero/one/few shot learning

Create a lot of chats and fine tune LLM to predict next word as it will be an conversation between 2 subjects



instruct GPT *fine tuning*

Prompt

Explain the moon landing to a 6 year old in a few sentences.

Completion

GPT-3

```
Explain the theory of gravity to a 6 year old.

Explain the theory of relativity to a 6 year old in a few sentences.

Explain the big bang theory to a 6 year old.

Explain evolution to a 6 year old.
```

InstructGPT

```
People went to the moon, and they took pictures of what they saw, and sent them back
to the earth so we could all see them.
```

chat GPT *fine tuning*

Hello, how are you → my

Hello, how are you my → friend

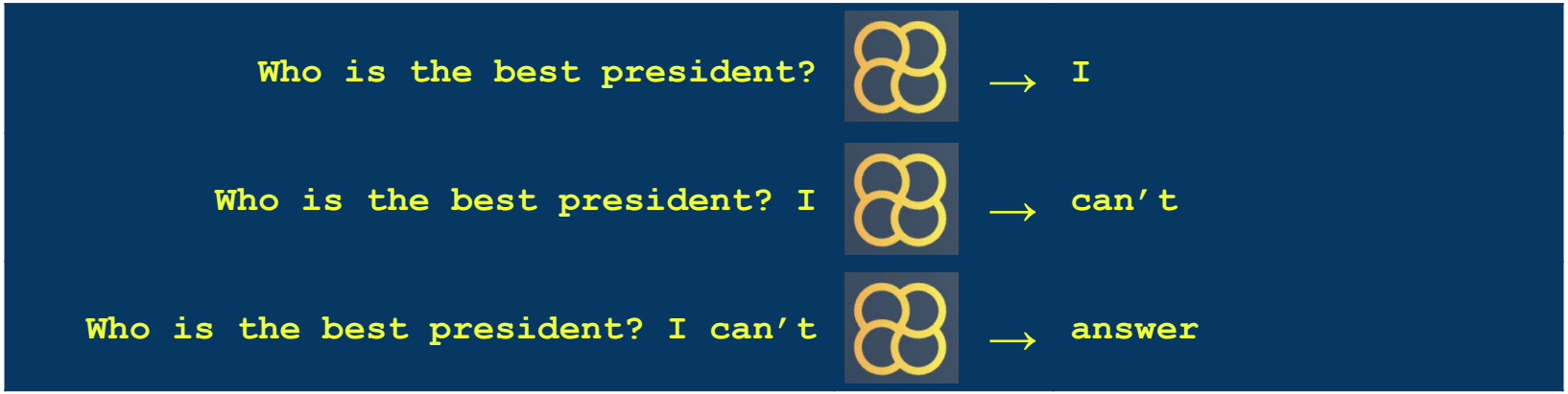
Hello, how are you my friend → ?

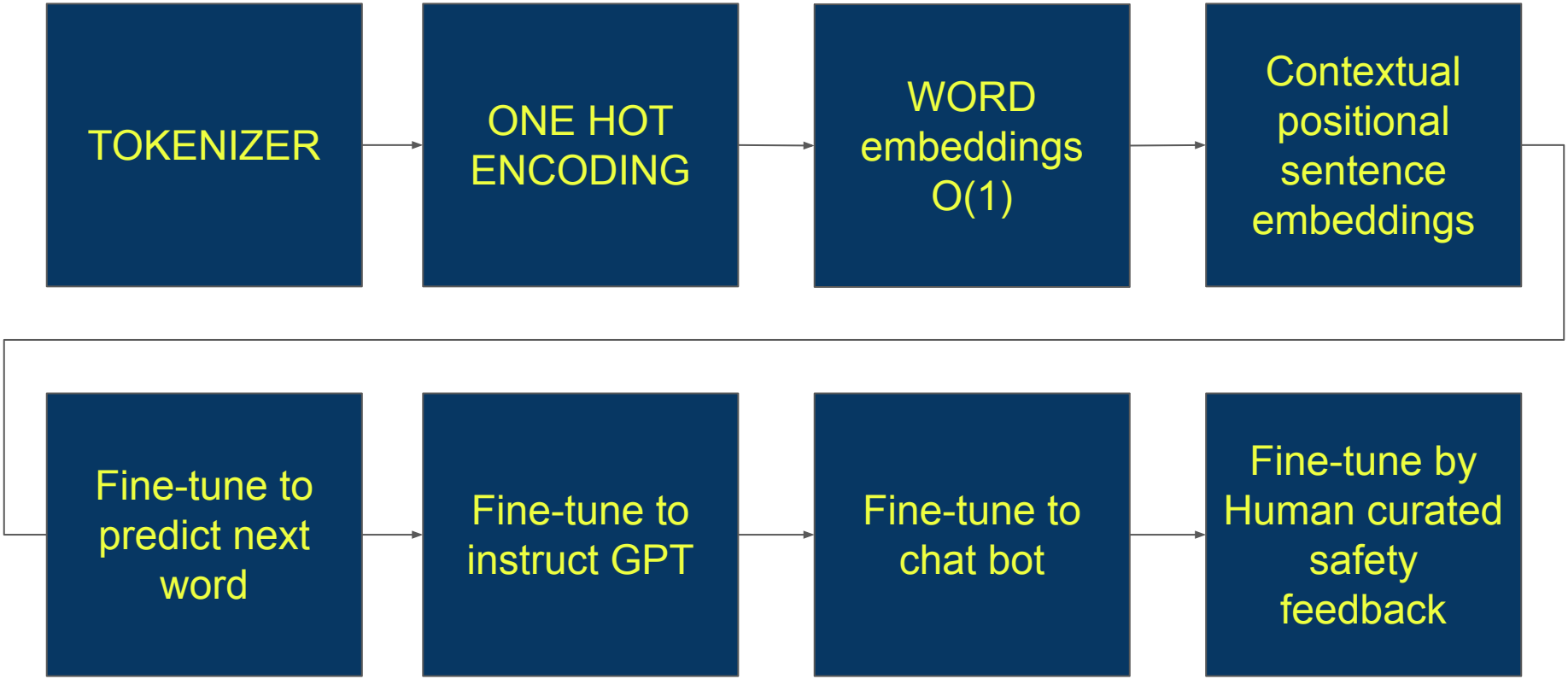
Hello, how are you → I'm

Hello, how are you I'am → Just

Hello, how are you I'm Just → LLM

Human-curated AI safety *fine tuning*

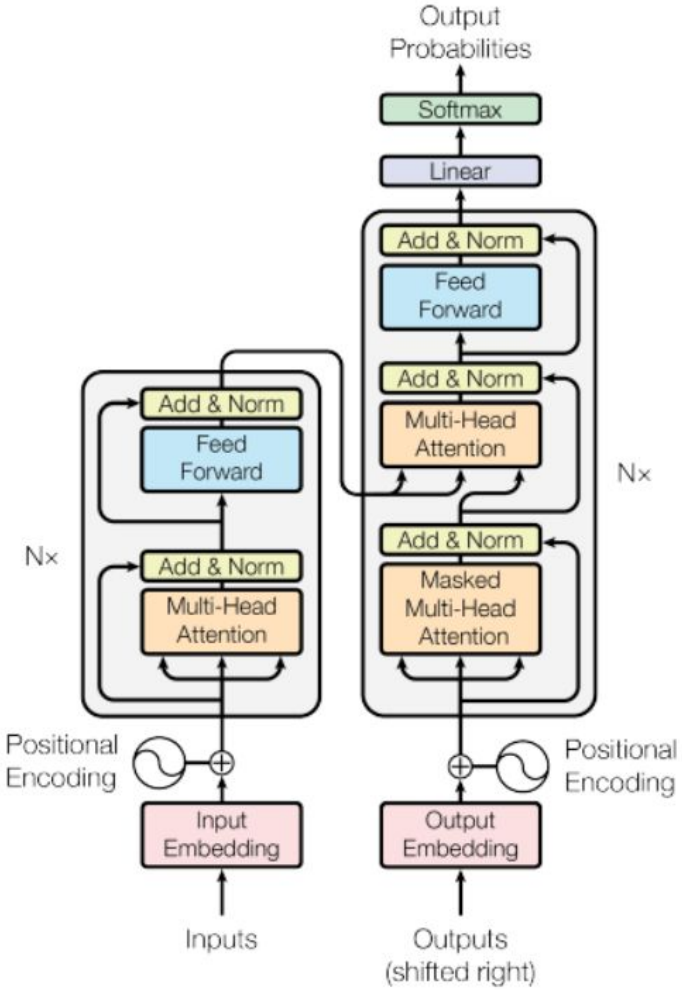



TOKENIZER
ONE HOT ENCODING
WORD embeddings O(1)
Contextual positional sentence embeddings
Fine-tune to predict next word
Fine-tune to instruct GPT
Fine-tune to chat bot
Fine-tune by Human curated safety feedback

chatLGPT → chatGPT

chatLGPT → chatGPT

- Make it larger
- Make it faster
- Make it more distributable
- Make it larger
- Make it faster
- Add gigant corpus



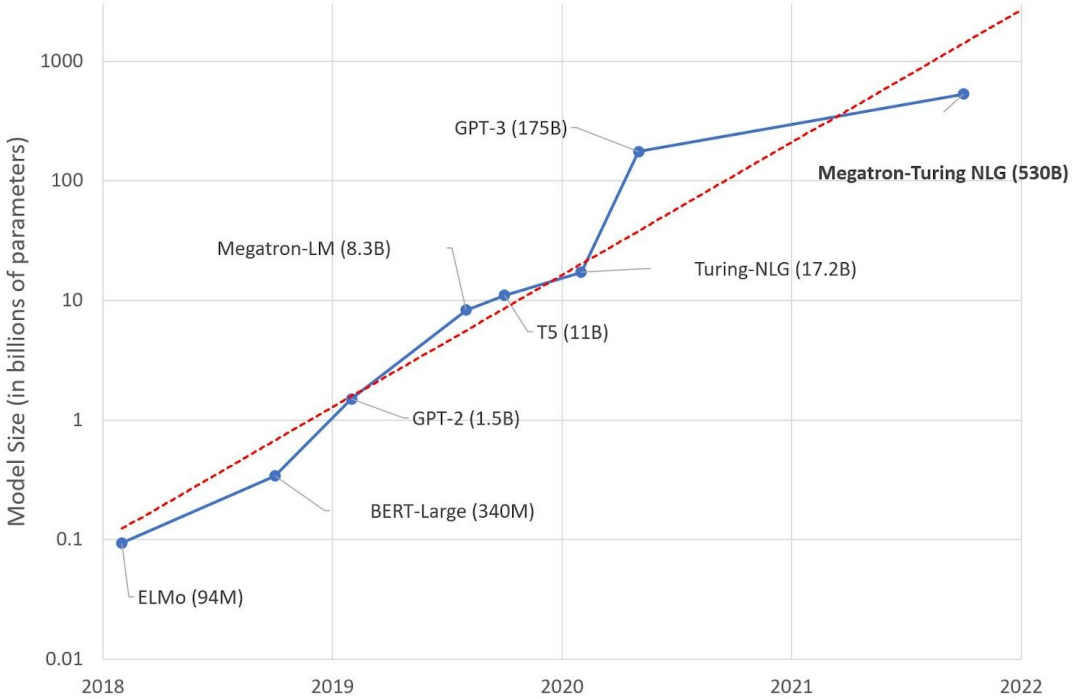
LLM comparison

LGPT 2023 (our)

- Corpus: 6 sentences
- Token count: 3
- parameters count: 14
- Window size: 2
- Embed vector dimension: 2

Open ai gpt3

- Corpus: multilingual internet + millions of books
- parameters count: 175B
- Window size: 2048
- Embed vector dimension: 12288



Mmm… who cares?

- LLM are “just” vector approximation machines
- LLM does not hold exact single source of truth
- LLM can’t keep factual database
- LLM are not the tools which solve 100% of your use cases
- Knowledge & halucinating of LLM is based on data which we use for learning

Current challenges in the field of NLP

- Short memory (people are trying to solve it with vector databases)
- O(N^2) memory consumption (window size)
- Gigantic hardware requirements
- Needs to scrape the whole internet to do just dump model
- hallucinations
- It works the same as human think => word by word till the book exist
- Many AI innovations are just how to optimise learning to be able to train the model in finite time on current hardware
- How to use GPT?
    - It’s not good to use it for fact-checking!!!
    - Its good for creative work
        - For example to summarize exact data which are factchecked like bing search do

Ezoteric future predictions

My humble future predictions

- FE UI will be dead => unified text interface to aggregate information across services (see bing chatbot)
- Code documentation like JSDoc is duplicity with the code itself
- Data types in programming are duplicates with runtime logic
- Documentation pages of programming libraries is duplicity with the code itself

- Natural languages gonna be less & less important because we have abstract vector
- LLM will be used as code generator for exact task where bugs are not allowed
- LLM will be used as interpreters for task which are not so exact

- Tutorials are dead, you only need to find a way how to put your raw data information into LLM
- Hard tradeoff between creating new nice API and generate old ugly but already included in the LLMs
- Monolits vs microservices => it depends on window-size of LLM => current SOTA LLM has around 1K lines=> it could generate whole microservice

END - Disclaimer

- Examples were simplified into small neural network with ~10 params
- Real LLM NLP neural nets has which I did not cover here:
  - Batch normalization
  - Multi head (self-)?Attention Layer
  - Positional encoding (i could add it into animation 🤔)
  - Residual connections (known from resNet)
  - Softmax (i could add it into animation 🤔)
  - dropout